# A Comparison of Software and Hardware Synchronization Mechanisms for Distributed Shared Memory Multiprocessors*

John B. Carter, Chen-Chi Kuo, Ravindra Kuramkote

{retrac, chenchi, kuramkot}@cs.utah.edu
WWW: http://www.cs.utah.edu/projects/avalanche

UUCS-96-011

Department of Computer Science
University of Utah, Salt Lake City, UT 84112

September 24, 1996

**Abstract**

Efficient synchronization is an essential component of parallel computing. The designers of traditional multiprocessors have included hardware support only for simple operations such as compare-and-swap and load-linked/store-conditional, while high level synchronization primitives such as locks, barriers, and condition variables have been implemented in software [9, 14, 15]. With the advent of directory-based distributed shared memory (DSM) multiprocessors with significant flexibility in their cache controllers [7, 12, 17], it is worthwhile considering whether this flexibility should be used to support higher level synchronization primitives in hardware. In particular, as part of maintaining data consistency, these architectures maintain lists of processors with a copy of a given cache line, which is most of the hardware needed to implement distributed locks.

We studied two software and four hardware implementations of locks and found that hardware implementation can reduce lock acquire and release times by 25-94% compared to well tuned software locks. In terms of macrobenchmark performance, hardware locks reduce application running times by up to 75% on a synthetic benchmark with heavy lock contention and by 3%-6% on a suite of SPLASH-2 benchmarks. In addition, emerging cache coherence protocols promise to increase the time spent synchronizing relative to the time spent accessing shared data, and our study shows that hardware locks can reduce SPLASH-2 execution times by up to 10-13% if the time spent accessing shared data is small.

Although the overall performance impact of hardware lock mechanisms varies tremendously depending on the application, the added hardware complexity on a flexible architecture like FLASH [12] or Avalanche [7] is negligible, and thus hardware support for high level synchronization operations should be provided.

*This work was supported by the Space and Naval Warfare Systems Command (SPAWAR) and Advanced Research Projects Agency (ARPA), Communication and Memory Architectures for Scalable Parallel Computing, ARPA order #B990 under SPAWAR contract #N00039-95-C-0018